INVESTOR IN PEOPLE

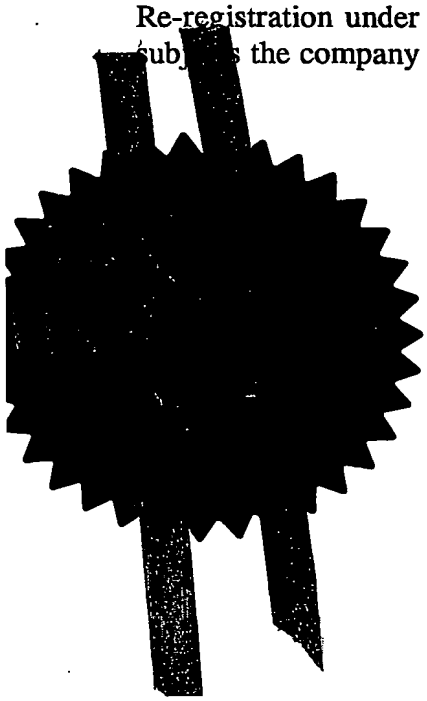
The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.



PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Signed

Dated 4 November 2003

Patents Form 1/77

Patents Act 1977
(Rule 16)

THE PATENT OFFICE
B
12 DEC 2002
LONDON

PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS
The Patent Office

13DEC02 E770768-1 002246
P01/7700 0.00-0229068.2

1/77

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference

P015724GB

2. Patent application number
(The Patent Office will fill in this part)

12 DEC 2002

0229068.2

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

ARM Limited
110 Fulbourn Road
Cherry Hinton
Cambridge
CB1 9NJ

07498124002

Patents ADP number *(if you know it)*

If the applicant is a corporate body, give the country/state of its incorporation

4. Title of the invention

Instruction Timing Control Within A Data Processing System

5. Name of your agent *(if you have one)*

D Young & Co

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

21 New Fetter Lane
London
EC4A 1DA

Patents ADP number *(if you know it)*

59006

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Yes

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	0
Description	10
Claim(s)	5
Abstract	1
Drawing(s)	2

10. If you are also filing any of the following, state how many against each item.

Priority documents	0
Translations of priority documents	0
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	2
Request for preliminary examination and search *(Patents Form 9/77)*	1
Request for substantive examination *(Patents Form 10/77)*	0
Any other documents *(please specify)*	0

11. I/We request the grant of a patent on the basis of this application.

Signature D Young & Co Date 12 December 2002

D Young & Co (Agents for the Applicants)

12. Name and daytime telephone number of person to contact in the United Kingdom — Nigel Robinson — 023 8071 9500

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

DYC Ref: P015724GB
ARM Ref: P249

INSTRUCTION TIMING CONTROL WITHIN A DATA PROCESSING SYSTEM

This invention relates to the field of data processing systems. More particularly, this invention relates to data processing systems in which at least some program instructions require a variable number of processing cycles to complete.

It is known to provide processor cores, such as those designed by ARM Limited of Cambridge, England, which have instructions requiring a variable number of processing cycles to complete. Some of these program instructions may be conditional instructions, which are encoded to only be executed if certain conditions are satisfied in relation to the execution of preceding instructions, e.g. a preceding instruction produced a zero result, an overflow, a carry or the like. In order to increase processor performance it is also known to provide processors with early termination techniques whereby a multicycle operation will be early terminated if the processor determines that the result will not be altered by subsequent processing cycles, e.g. a multicycle add operation in which all of the high order bits of both operands are zeros and so there is no need to calculate the high order bits in the later processor cycles once it has been determined that there is no carry into them. There are many other examples of program instructions which can take a variable number of processing cycles to complete.

It is becoming increasingly important to provide processors which can operate with a high degree of security, for example, such that the data which they are manipulating cannot be subject to unauthorised access. As an example, processor cores provided in smartcards for handling financial transactions often utilise highly secret cryptographic keys to verify passwords, data and the like. These cryptographic keys must be kept secret if the security and integrity of the system is to be maintained. One known technique for attacking the security of such systems is to try to identify information regarding the data being processed by counting the number of processing cycles taken to complete a given operation. If there is a correlation between the number of processing cycles taken and the data being processed, then it is possible for this to reveal information regarding the data being processed.

It is known to seek to resist cycle counting attacks on the security of a system by carefully writing the computer program code such that it will take a fixed number of cycles to complete irrespective of the flow through the computer program. In practice this is difficult to achieve since the knowledge and skill required by a programmer is of a high level and it is difficult to ensure/test that the desired aim of making the number of processing cycles equal in all circumstances has been achieved. Furthermore, many of the productivity enhancing tools commonly used by programmers, such as compilers, are not well suited to generating such secure code since they will normally produce code which varies the number of processing cycles consumed in a significant way depending upon the program path being followed.

Viewed from one aspect the present invention provides apparatus for processing data, said apparatus comprising:

a processor responsive to a plurality of different program instructions to perform respective processing operations each requiring a number of processing cycles to complete, said plurality of program instructions including at least one variable timing instruction requiring between a minimum number of cycles and a maximum number of cycles to complete, wherein

said processor is operable in a variable timing mode in which said at least one variable timing instruction is permitted to take a variable number of processing cycles to complete; and

said processor is operable in a fixed timing mode in which said at least one of variable timing instruction is forced to take said maximum number of cycles to complete.

The present invention recognises that the variable timing instructions provided within processors to speed processing operation and reduce power consumption can be a positive disadvantage in the context of a secure system in which it is desired to obscure any relationship between the number of processing cycles taken and the data being processed. At the same time, the strong advantages associated with variable timing instructions, such as increased speed and reduced power consumption are desirable when not processing secure data. The present technique recognises the above problem and also maintains the advantages associated with the existing systems in those circumstances where this is possible by providing a system operable in a

variable timing mode and a fixed timing mode. In the variable timing mode, variable timing instructions are permitted to take their variable number of processing cycles to complete whereas in the fixed timing mode the variable timing instructions are constrained to always take a fixed number of cycles to complete. Thus, when processing secure data, the system may be placed in the fixed timing mode and the number of processing cycles taken to complete a processing operation more readily obscured whereas when not processing secure data the system may operate in the variable timing mode whereby the speed advantages and reduced power consumption advantages associated with the mechanisms for variable timing are exploited. The fixed timing mode makes it easier for a programmer to write program code which will take a fixed number of processing cycles to complete irrespective of the program flow path since the timing of each instruction will be independent of the data being processed. This also makes it easier to provide compilers able to generate fixed timing code.

As previously mentioned, the present technique is applicable to a wide range of program instructions that can take variable timings, but is particularly well suited to embodiments in which one or more conditional instructions is executed in a manner dependent upon whether the system is in the fixed timing mode in which the conditional instruction is forced to take a fixed number of processing cycles irrespective of any condition codes and a variable timing mode in which the execution of the instruction may be suppressed if the condition codes match certain parameters.

In the context of executing conditional instructions in the fixed timing mode that would have been suppressed in the variable timing mode, preferred embodiments of the invention are such that the conditional instruction is blocked from making any change having an effect upon subsequent data processing, e.g. the conditional instruction may be prevented from updating or changing any state within the processor which could influence subsequent processing.

A type of program instruction to which the present technique is well suited is a conditional branch instruction as without this technique such instructions can have a strong influence upon program timing depending upon whether or not the branch is suppressed. When operating in the fixed timing mode, a conditional branch

instruction will be forced to perform a branch even when the condition codes are not met, but in this circumstance the branch will be to the next instruction in the program flow irrespective of the branch target specified in the conditional branch instruction such that the next instruction is executed in the same way as if the conditional branch instruction had been suppressed in its entirety.

In the case where the conditional instruction is a conditional data manipulating instruction, then preferred embodiments of the invention are such that when in the fixed timing mode the conditional data manipulating instruction is executed when the condition codes are failed and it would be normally suppressed such that the instruction is executed but is prevented from writing a result in any normal result destination of the conditional data manipulating instruction.

In order to resist a security attack based upon observing a variation in the power consumed by the processor dependent upon whether or not a result is being written from a conditional data manipulating instruction, preferred embodiments operate such that a result is written to one or more dummy destinations when the condition codes are failed instead of being written to the normal result destination. In this way, the power consumption can be kept substantially the same irrespective of whether or not the condition codes are failed with the conditional data manipulating instruction always being executed and always resulting in a result being written even if this is only to a dummy destination.

Another type of instructions capable of variable timing execution are those capable of early termination in dependence upon one or more data values being processed. Preferred embodiments operate in the fixed timing mode to suppress any early termination even when the mechanisms in place within the system for providing such early termination variable timing mode indicate that it is possible. As an example, it may be detected that one of the operands in a multiply is a zero and accordingly the result will be a zero irrespective of the other operands and so the instruction is capable of being early terminated with a result of zero. However, in the fixed timing mode, the instruction will be forced to utilise its full number of processing cycles and then return the zero result.

As examples of the preferred types of instruction capable of early termination there are multicycle multiply instructions, multicycle divide instructions, multicycle add instructions and multicycle subtract instructions.

Whilst it is possible that the system may be placed into the fixed timing mode in dependence upon some hardware signal or the like, in preferred embodiments of the invention the processor may be configured to adopt either the variable timing mode or the fixed timing mode in dependence upon a programmable mode controlling parameter, e.g. one or several bits.

In this way, the system can be switched under software control between the variable timing mode when secure data is not being processed and high performance is required with low power consumption and a fixed timing mode when secure data is being processed and the requirement to obscure any dependence upon the data being processed of the number of processing cycles take takes precedence over speed or power consumption considerations.

A preferred way of setting the programmable mode controlling parameter is to store this within a system configuration register, such as a system controlling coprocessor.

Viewed from another aspect the present invention provides a method of processing data using a processor responsive to a plurality of different program instructions to perform respective processing operations each requiring a number of processing cycles to complete, said plurality of program instructions including at least one variable timing instruction requiring between a minimum number of cycles and a maximum number of cycles to complete, said method comprising the steps of:

operating said processor in a variable timing mode in which said at least one variable timing instruction is permitted to take a variable number of processing cycles to complete; and

operating said processor in a fixed timing mode in which said at least one of variable timing instruction is forced to take said maximum number of cycles to complete.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 schematically illustrates a data processing system operable in a fixed timing mode and a variable timing mode.

Figure 2 schematically illustrates a conditional program instruction;

Figure 3 is a flow diagram schematically illustrating part of the processing operations performed by an instruction decoder operating in accordance with the present technique and;

Figure 4 schematically illustrates the execution of a conditional branch instruction in a fixed timing mode.

Figure 1 illustrates a data processing system 2 including a processor core 4, a coprocessor 6 and a memory 8.

In operation, the processor core 4 fetches instructions and data from the memory 8. The instructions are fed to an instruction pipeline 10 where they occupy successive pipeline stages such as, for example, fetch, decode, execute, memory and write back on successive processing cycles. Pipelined processors are in themselves well known as a way of effectively executing a number of program instructions in a partially overlapping fashion in order to improve processor performance.

The data values read from the memory 8 by the processor core 4 are supplied to a register bank 12 from where they may be manipulated under program instruction control using one or more of a multiplier 14, a shifter 16 and an adder 18. Other data manipulating circuits may be provided, such as circuits performing logical operations, such as ANDs, Ors, count leading zeros etc.

Figure 1 also illustrates an instruction decoder 20 within the processor core 4 which is responsive to a program instruction within the instruction pipeline 10 to generate execution control signals that are applied to the various processing elements,

such as the register bank 12, the multiplier 14, the shifter 16 and the adder 18 in order to control the data processing operations performed. As an example, the control signals generated by the decoder 20 may cause the appropriate operands to be read from the register bank 12 and supplied and acted upon by the appropriate ones of the multiplier 14, the shifter 16 and the adder 18 so as to generate a result which is written back into the register bank 12.

The coprocessor 6 is a system configuration coprocessor containing a number of configuration registers 22 which may be written under program control to set up configuration controlling parameters. These configuration controlling parameters can specify many aspects of the configuration of the processing system 2, such as for example the endianess and the like. Included within one of these configuration controlling registers 22 is a bit which specifies whether or not the processor core should operate in a fixed timing mode or a variable timing mode. This bit is illustrated as being supplied as an input to the instruction decoder 20, but it will be appreciated that this bit may also be supplied to various other points within the processor core 4 as required to control their behaviour. In dependence upon this fixed/variable bit, the processor core 4 operates in either a fixed timing mode or a variable timing mode. When in the fixed timing mode at least one program instruction which has a variable timing (i.e. takes a variable number of processing cycles to complete) in the variable timing mode, is instead forced to have a fixed timing (e.g. take the maximum possible number of processing cycles to complete irrespective of whether or not it could have been suppressed in its entirety or completed in less than the maximum number of processing cycles. As the instruction decoder 20 is primarily responsible for decoding the program instructions and instructing the activity of the other elements of the processor core 4, the instruction decoder 20 can take the major role in controlling the processor core 4 to either operate in the fixed timing mode or the variable timing mode. Not all variable timing instruction need be provided with a fixed timing mode type of operation.

It will be appreciated that in the above description a single bit in the configuration controlling register 22 is shown as switching between fixed and variable timing modes. Alternatively, multiple bits within the configuration controlling register 22 may be provided to separately enable and disable the fixed or variable timing

behaviour of different types of instruction, such as conditional instruction behaviour, uniform branch behaviour, disabling early terminate, etc.

Figure 2 schematically illustrates a conditional instruction 24. This conditional instruction may be part of an instruction set which includes only some conditional instructions or part of an instruction set, such as the ARM instruction set, which is substantially fully conditional. The condition codes 26 encode a set of processor state conditions in which the associated instruction either will or will not be executed. As an example, the condition codes 26 can be arranged to specify that the instruction 24 will not execute if the condition codes currently set in the system indicate a zero result, a carry has occurred, an overflow has occurred or the like. This type of instruction can be utilised to provide efficient program coding. The fixed/variable bit at least partially suppresses the conditional behaviour in that the instruction will execute irrespective of its condition codes, but may not write its result in a way that has an effect upon the processor state.

Figure 3 is a flow diagram schematically illustrating part of the processing operations performed by the instruction decoder 20. It will be appreciated that Figure 3 illustrates these processing operations as a logical sequence, whereas in practice these processing operations may be performed at least partially in parallel or in a different order.

At step 28, the instruction decoder 20 waits for a new instruction to execute. When a new instruction is received processing proceeds to step 30 at which the condition codes associated with the new instruction are read. At step 32 these condition codes are compared with the currently existing condition codes in the system. These condition codes currently existing in the system are the result of previous processing activity, either in the immediately preceding instruction or in the last instruction which would have updated those condition codes.

At step 34, a check is made for a match between the condition codes 26 of the current instruction being executed and the existing condition codes. If a match does not occur, then processing proceeds to step 36 where execution of the current instruction is started. It will be appreciated that Figure 3 illustrates a system in which

execution occurs when a match does not occur, but alternative embodiments could equally well be ones in which execution occurs when a match occurs.

Following step 36, processing proceeds to step 38 where a check is made as to whether or not early termination of the instruction is possible. This early termination may, for example, be because one of the operands has a particular value, such as zero or unity, or on subsequent processing cycles that a particular partial result has been produced. If early termination is possible, then processing proceeds to step 40 where a check is made as to whether or not the processor core 4 is currently operating in the fixed or variable timing mode. If the processor is in the variable timing mode, then processing proceeds to step 42 and the instruction concerned is early terminated with the result being returned as appropriate and processing returns to step 28.

If the determination at step 40 is that the system is in the fixed timing mode, then processing proceeds to step 44 irrespective of the fact that early termination is possible. Step 44, which may also be reached by a determination at step 38 that early termination is not possible, executes the instruction concerned for one processing cycle. In the case of a multicycle processing instruction, such as a multiplication, a divide, an add or a subtraction, these typically take several cycles to execute and so after step 44 processing proceeds to step 46 at which a determination is made as to whether or not the maximum number of cycles associated with that instruction has yet been performed. If the maximum number of cycles has been performed, then the result will have been generated. If early termination was possible and the system was being forced to continue to execute for further processing cycles, then step 46 will still indicate that this forced execution should cease when the maximum possible number of processing cycles for that type of instruction has been reached. If the maximum number of processing cycles has not yet been performed, then processing is returned to step 38.

If the match tested for at step 34 was positive, then processing proceeds to step 48. In this example, the positive detection of a match at step 34 indicates that execution of the particular instruction should be suppressed. Step 48 determines whether or not the system is currently in the fixed timing mode. If in the fixed timing mode, then processing proceeds to step 50 where a forced dummy execution of the

instruction will occur. When dummy execution is performed the result is written to a dummy register (see dummy register 51 in Figure 1), rather than the destination specified in the instruction itself so as to prevent the state of the system being modified by a program instruction which should not have executed as it should have been suppressed whilst also keeping a substantially unaltered power consumption. If at step 48 the determination is that the system is not in the fixed timing mode but is in the variable timing mode, then processing bypasses step 50 and returns to step 28 with the program instruction being suppressed in the normal way.

It will be appreciated that Figure 3 illustrates a generic system in which dummy execution is applied to all condition code failed instructions and all early termination of instructions is suppressed. In practice, it is also possible for these techniques to be applied to a subset of conditional instructions and instructions capable of early termination. The multiple configuration controlling bits mentioned above could be used to selectively turn on features such as early terminate suppression, but not others, such as dummy execution following a condition code fail.

Figure 4 schematically illustrates the execution of a conditional branch instruction in the fixed timing mode. A sequence of instructions AB are executed until a conditional branch instruction BEQ (branch upon equal) is reached. This instruction encodes the behaviour that the specified branch will be performed if the flag indicating an equal result from previous processing is set and will be suppressed if this flag is not set. When the condition codes are passed, i.e. a condition code match, then the branch is taken and processing proceeds to instructions X, Y, etc. If the condition codes fail, then instead of being suppressed in its entirety, the BEQ instruction performs a branch to the immediately following instruction C. This is the same instruction which would have been reached if the BEQ instruction had been suppressed and not executed at all. However, in the fixed timing mode, the BEQ will have executed consuming the same number of processing cycles irrespective of whether or not the condition codes were passed or failed. This helps obscure the results of data processing operations previously performed from a person trying to gain access to secure data.

CLAIMS

1. Apparatus for processing data, said apparatus comprising:

a processor responsive to a plurality of different program instructions to perform respective processing operations each requiring a number of processing cycles to complete, said plurality of program instructions including at least one variable timing instruction requiring between a minimum number of cycles and a maximum number of cycles to complete, wherein

said processor is operable in a variable timing mode in which said at least one variable timing instruction is permitted to take a variable number of processing cycles to complete; and

said processor is operable in a fixed timing mode in which said at least one of variable timing instruction is forced to take said maximum number of cycles to complete.

2. Apparatus as claimed in claim 1, wherein said at least one variable timing instruction includes a conditional instruction, said processor being operable in said variable timing mode to suppress execution of said conditional instruction in dependence upon one or more condition codes set in response to execution of one or more previously executed program instructions and said processor being operable in said fixed timing mode to complete said conditional instruction in a fixed number of processing cycles irrespective of said one or more condition codes set in response to execution of one or more previously executed program instructions.

3. Apparatus as claimed in claim 2, wherein when executing said conditional instruction in said fixed timing mode, if said conditional codes are such that execution of said conditional instruction would have been suppressed in said variable timing mode, then said conditional instruction is blocked for making any change effecting subsequent data processing operations.

4. Apparatus as claimed in claim 3, wherein if said conditional instruction is a conditional branch instruction, then when executing said conditional branch instruction in said fixed timing mode, if said conditional codes are such that execution of said conditional branch instruction would have been suppressed in said variable

timing mode, then said conditional branch instruction is forced to perform a branch to a next instruction irrespective of any branch target specified in said conditional branch instruction.

5. Apparatus as claimed in claim 3, wherein if said conditional instruction is a conditional data manipulating instruction, then when executing said conditional data manipulating instruction in said fixed timing mode, if said conditional codes are such that execution of said conditional data manipulating instruction would have been suppressed in said variable timing mode, then said conditional data manipulating instruction is prevented from writing a result to any normal result destination of said conditional data manipulating instruction.

6. Apparatus as claimed in claim 5, wherein if writing of said result to any normal destination is prevented, then said result is instead written to at least one dummy destination.

7. Apparatus as claimed in claim 6, wherein said at least one dummy destination include a dummy processor register.

8. Apparatus as claimed in claim 1, wherein said at least one variable timing instruction includes an instruction capable of early termination in dependence upon one or more data values being processed, said processor being operable in said variable timing mode to permit early termination and said processor being operable in said fixed timing mode to prevent early termination.

9. Apparatus as claimed in claim 8, wherein said instruction capable of early termination is one of:

a multicycle multiply instruction;

a multicycle divide instruction;

a multicycle add instruction; and

a mutlicycle subtract instruction.

10. Apparatus as claimed in any one of the preceding claims, wherein said processor adopts said variable timing mode or said fixed timing mode in dependence upon a programmable mode controlling parameter.

11. Apparatus as claimed in claim 10, wherein said programmable mode controlling parameter is stored within a system configuration register.

12. Apparatus as claimed in any one of the preceding claims, wherein said processor is switched into said fixed timing mode so as to disguise a program execution path.

13. A method of processing data using a processor responsive to a plurality of different program instructions to perform respective processing operations each requiring a number of processing cycles to complete, said plurality of program instructions including at least one variable timing instruction requiring between a minimum number of cycles and a maximum number of cycles to complete, said method comprising the steps of:

operating said processor in a variable timing mode in which said at least one variable timing instruction is permitted to take a variable number of processing cycles to complete; and

operating said processor in a fixed timing mode in which said at least one of variable timing instruction is forced to take said maximum number of cycles to complete.

14. A method as claimed in claim 13, wherein said at least one variable timing instruction includes a conditional instruction and further comprising operating said processor in said variable timing mode to suppress execution of said conditional instruction in dependence upon one or more condition codes set in response to execution of one or more previously executed program instructions and said processor being operable in said fixed timing mode to complete said conditional instruction in a fixed number of processing cycles irrespective of said one or more condition codes set in response to execution of one or more previously executed program instructions.

15. A method as claimed in claim 14, wherein when executing said conditional instruction in said fixed timing mode, if said conditional codes are such that execution of said conditional instruction would have been suppressed in said variable timing mode, then said conditional instruction is blocked for making any change effecting subsequent data processing operations.

16. A method as claimed in claim 15, wherein if said conditional instruction is a conditional branch instruction, then when executing said conditional branch instruction in said fixed timing mode, if said conditional codes are such that execution of said conditional branch instruction would have been suppressed in said variable timing mode, then said conditional branch instruction is forced to perform a branch to a next instruction irrespective of any branch target specified in said conditional branch instruction.

17. A method as claimed in claim 15, wherein if said conditional instruction is a conditional data manipulating instruction, then when executing said conditional data manipulating instruction in said fixed timing mode, if said conditional codes are such that execution of said conditional data manipulating instruction would have been suppressed in said variable timing mode, then said conditional data manipulating instruction is prevented from writing a result to any normal result destination of said conditional data manipulating instruction.

18. A method as claimed in claim 17, wherein if writing of said result to any normal destination is prevented, then said result is instead written to at least one dummy destination.

19. A method as claimed in claim 18, wherein said at least one dummy destination include a dummy processor register.

20. A method as claimed in claim 13, wherein said at least one variable timing instruction includes an instruction capable of early termination in dependence upon one or more data values being processed and further comprising operating said processor in said variable timing mode to permit early termination and said processor being operable in said fixed timing mode to prevent early termination.

21. A method as claimed in claim 20, wherein said instruction capable of early termination is one of:

a multicycle multiply instruction;

a multicycle divide instruction;

a multicycle add instruction; and

a mutlicycle substract instruction.

22. A method as claimed in any one of claims 13 to 21, wherein said processor adopts said variable timing mode or said fixed timing mode in dependence upon a programmable mode controlling parameter.

23. A method as claimed in claim 22, wherein said programmable mode controlling parameter is stored within a system configuration register.

24. A method as claimed in any one of claims 13 to 23, wherein said processor is switched into said fixed timing mode so as to disguise a program execution path.

ABSTRACT

INSTRUCTION TIMING CONTROL WITHIN A DATA PROCESSING SYSTEM

A data processing system 2 is provided which is responsive to program instructions that operate in a variable timing mode to require a variable number of processing cycles to complete. The system is also operable in a fixed timing mode, which may be programmable using a bit (or several bits) within a configuration controlling register, to operate in a fixed timing mode in which such instructions are forced to operate using a fixed number of processing cycles. Thus, suppression of instructions which fail their condition codes may be suppressed and early termination of program instructions similarly suppressed in a manner which helps resist an attack upon the security of the system by observing the number of processing cycles required to process certain data.

[Figure 3]

Conditional Branch
in Fixed Timing Mode

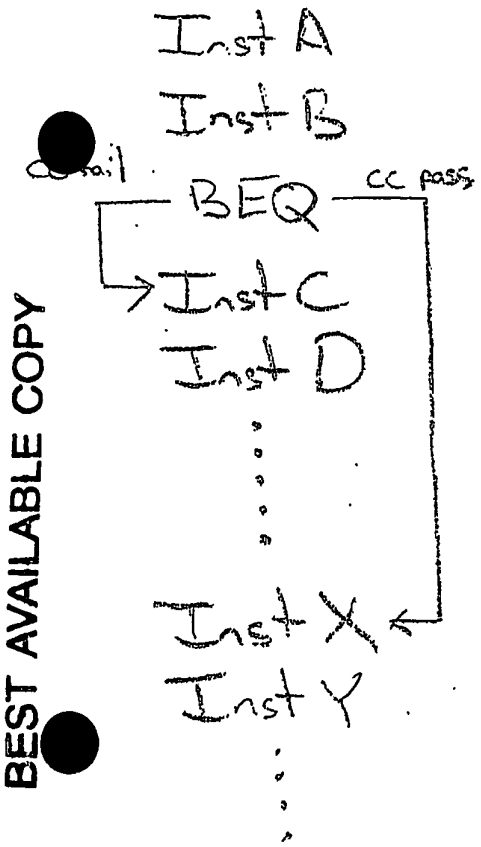


Fig. 4

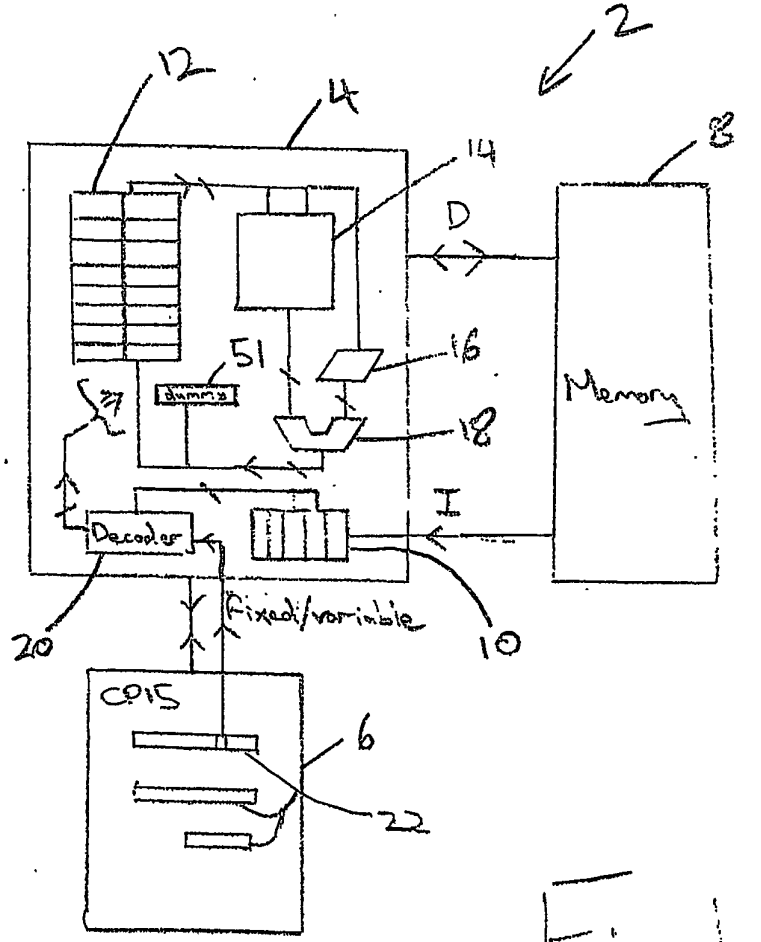


Fig. 1

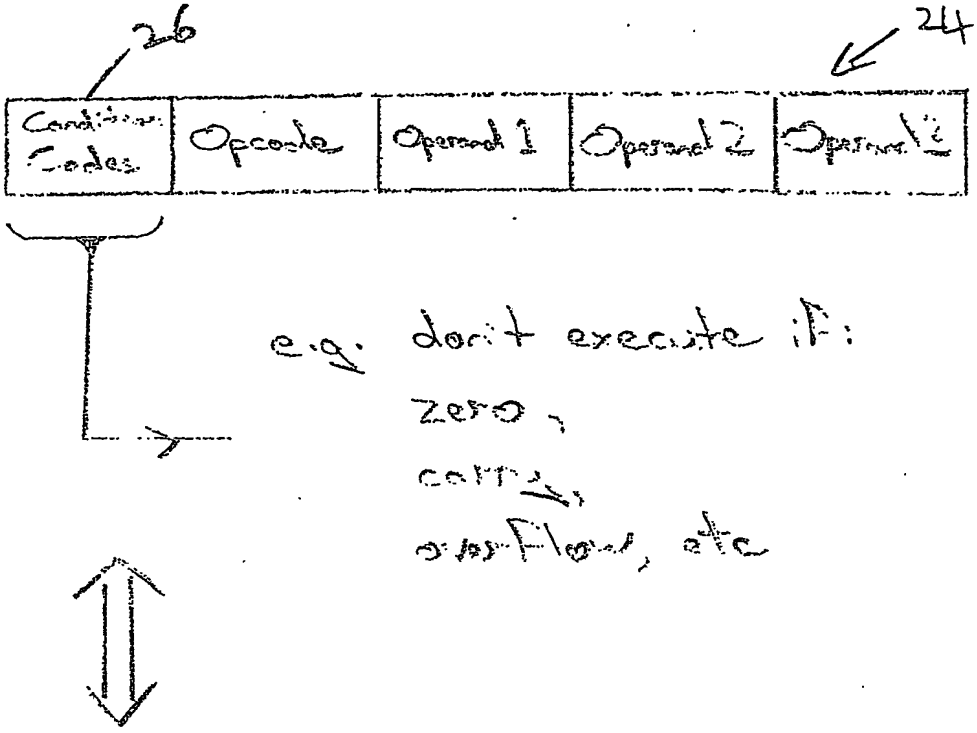


Fig. 2

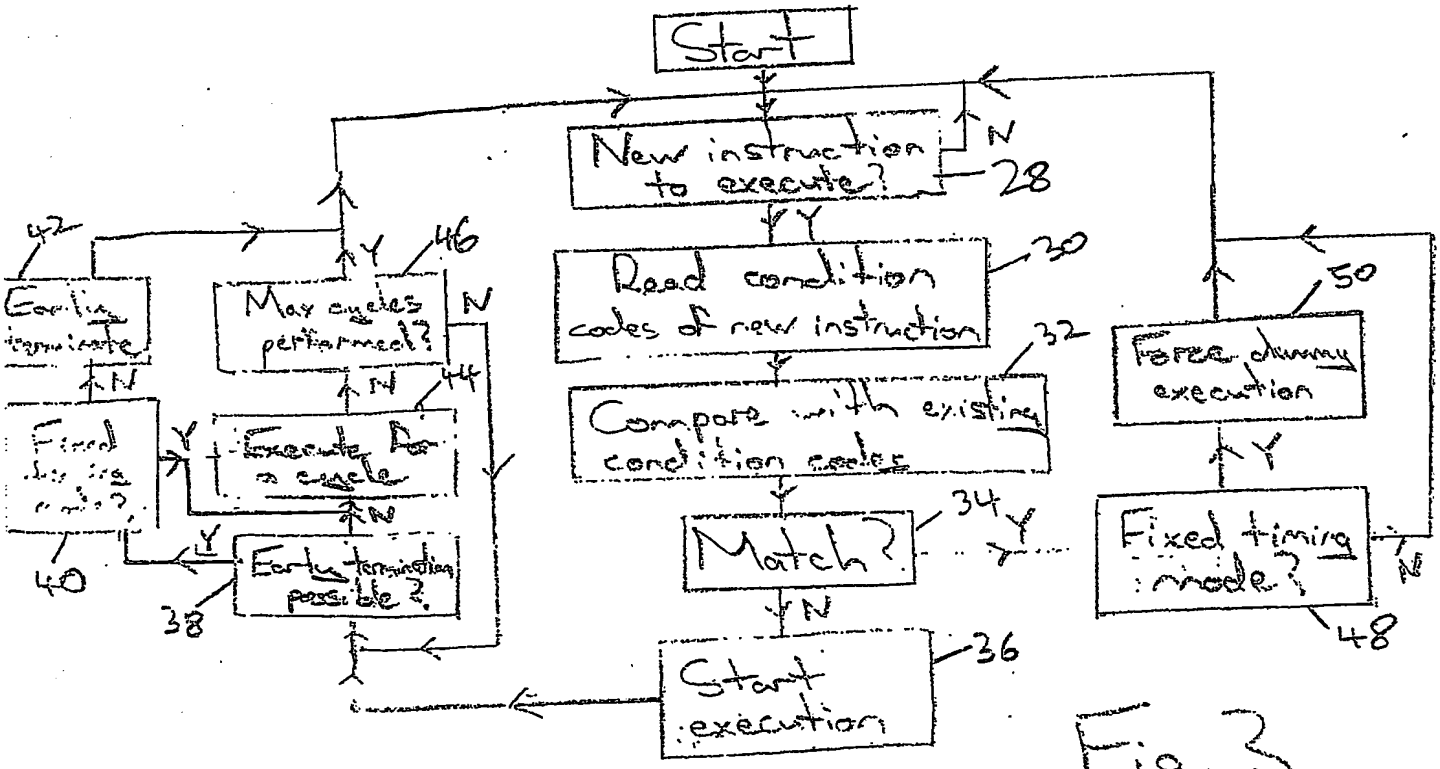


Fig. 3